IN THE SPECIFICATION:

Please amend the Specification of the above-identified application as follows.

Please amend the paragraph beginning on page 2, line 8 as follows.

-- There are currently several commercially available software programs, such as ~~ZoneAlarm Pro~~ZONEALARM PRO® manufactured by ZoneLabs, San Francisco, CA, ~~McAfee Firewall~~MCAFEE FIREWALL® manufactured by ~~Network Associates~~NETWORK ASSOCIATES®, Inc., Santa Clara, California, ~~Norton Internet Security~~NORTON INTERNET SECURITY 2002® manufactured by ~~Symantec Corp.~~SYMANTEC CORP.®, Cupertino, CA, ~~Norton Personal Firewall~~NORTON PERSONAL FIREWALL 2002® manufactured by ~~Symantec Corp~~SYMANTEC CORP.®, Cupertino, CA and ~~BlackIce Defender~~BLACKICE DEFENDER® manufactured by ~~Defender Network ICE Corporation~~DEFENDER NETWORK ICE CORPORATION®, San Mateo, CA, that place a firewall between the computer and the insecure network. In particular, the ~~ZoneAlarm~~ZONEALARM® program allows users to decide which applications can and cannot use the Internet. An Internet Lock is implemented in the ~~ZoneAlarm~~ZONEALARM® program for blocking Internet traffic while the computer is unattended or while the Internet is not being used. The ~~McAfee firewall~~MCAFEE FIREWALL® program, on the other hand, filters all the applications, system services, and protocols, including file and printer

shares (NetBIOS), IP protocols (TCP/IP, UDP/IP), service-based protocols (FTP, Telnet), ARP/RARP, and Dynamic Host Configuration Protocol (DHCP). Additionally, the firewall blocks the IPX and the NetBEUI on a per device basis.--

Please amend the paragraph beginning on page 3, line 1 as follows.

-- The ~~Norton Internet Security~~NORTON INTERNET SECURITY® 2002 program and ~~Norton Personal Firewall~~NORTON PERSONAL FIREWALL® 2002 program offers a software program that blocks incoming hack attacks while allowing trusted applications to connect to the computer. Lastly, the ~~BlackIce Defender~~BLACKICE DEFENDER® scans the DSL, cable modem or dial-up Internet connection for hacker activity. When an attempted intrusion is detected, the traffic from that source will be automatically blocked. As a result, any unwanted intrusion is avoided. In all these examples, the connection between the computer and the insecure network remains connected. Basically, all of the prior solutions filter the connection to the insecure network. In other words, while the computer is connected to the insecure network, the known programs provide a security system in front of the gateways or ports to the computer. The programs determine whether a requesting source is trusted or untrusted, and only the trusted sources are allowed access to the gateway or the ports.--

Please amend the paragraph beginning on page 4, line 14 as follows.

-- Also, in another embodiment, the present invention provides a computer program product ~~comprising~~including a computer readable code stored on a computer readable medium that, when executed, the computer program product causes a computer to determine whether the computer is active, deactivate the computer from the insecure network when it is determined that the computer is inactive, and wait for a predefined time period to repeat the method.--

Please amend the paragraph beginning on page 6, line 3 as follows.

-- A schematic diagram of a network system is shown in FIG. 1, and indicated generally at 10. A computer 12 is shown to be connected to the Internet 14 (i.e., insecure network) and a LAN 16 (secure network) running an intranet via a computer server 18. As shown, there are multiple computers 20, 22, 24, 26 including the computer 12, which are referred to as client computers, connected to the computer server ~~computer~~ 18. The Internet 14 also shows multiple computers 28, 30, 32, 34, 36, 38, 40 including the computer 12. However, in practice, the Internet 14 generally includes millions of computers connected at any given time, but, for simplicity, only 8 computers are shown. As a result of these various unidentified computers connected to the Internet 14, the computer 12 is very vulnerable to unwanted connections, such as from hackers or transmitters of potentially disabling computer viruses.--

Please amend the paragraph beginning on page 6, line 16 as follows.

-- Although the insecure network ~~shown~~ 10 is preferably connected to the Internet 14, other types of networks can certainly be used in conjunction with the Internet or even in place of it. For example, the network connection may include other Wide Area Networks (WANs) or even LANs. The present invention can be implemented with any type of network that is considered insecure, and these other implementations should be apparent to one skilled in the art.--

Please amend the paragraph beginning on page 6, line 23 as follows.

-- However, because the network system 10 is contemplated as varying greatly in types, complexity and size, an explanation of the current preferred embodiment of the network topology is given for clarification purposes. Thus, simply as an example, a computer 12 installed with the ~~Microsoft~~MICROSOFT® ~~Windows~~WINDOWS® operating system having a continuous connection to the Internet (i.e., insecure network) will be used as an example in describing one implementation of the present invention. However, other implementations with different software programs, such as network security programs, network programs or operating systems, are contemplated, and they are considered to be within the scope of the present invention.--

Please amend the paragraph beginning on page 7, line 22 as follows.

-- Turning to FIG. 3, the first step in the ~~Windows~~WINDOWS® environment is to first initialize the ~~Windows~~WINDOWS® sockets support or driver (block 56), followed by a step of loading a "INETMIB1.DLL" file or driver (block 58). After this, two addresses for two functions of SNMPEXTENSIONINIT and SNMPEXTENSIONQUERY are obtained from the INETMIB1.DLL (block 60). The SNMPEXTENSIONINIT function is then called in order to initialize the INETMIB1.DLL file (block 62). After the INETMIB1.DLL is initialized (block 62), the address (e.g., the object identifier) of a network card (e.g., 1.3.6.1.2.1.2.1.0) is now obtained (block 64). Next, the number of the interface(s) at the address of the network card (e.g., 1.3.6.1.2.1.2.0) is read from the INETMIB1.DLL file (block 66) and stored in memory. The status of the interface is also read at this time at the address or object identifier of the interface (e.g., 1.3.6.1.2.1.2.7.?) (block 68). Note that a question mark (?) has been used to indicate the address of the interface, because the actual address is not known, since the address of the interface is a variable generated at the time when the connection is made. Once all the information is obtained, the last status and the address/object identifier of the interface is then saved into memory (block 70).--

Please amend the paragraph beginning on page 8, line 15 as follows.

-- Turning back to FIG. 2, after the address of the network card and the interface connected to the insecure network along with its status are obtained (block 54), the next step is to wait for a predefined time period (block 72), which can be implemented according to the computer engineers' desire. Nevertheless, the time out period is preferably less than 30 seconds in order to ensure that the computer is constantly checked for deactivation from the insecure network. After waiting for the predefined time out period (block ~~74~~72), the method 50 will then determine whether there is a network reactivation request (block 74). In the present invention, this command is preferably requested through a user interface by users, but it is also contemplated that other programs in the system may request the network reactivation as well. For example, when a program installed on the computer makes a request to utilize the insecure network, a command to reactivate the network can be generated automatically in the present invention. As a result, even if the user does not directly request the reactivation, it is contemplated that other programs, nevertheless, can trigger the reactivation or deactivation of the insecure network in the present invention. Again, these other various implementations are within the scope of the present invention.--

Please amend the paragraph beginning on page 9, line 8 as follows.

-- If there is a request to reactivate the network (block 74), ~~the~~a subroutine for reactivating the network (block 76) in the ~~Windows~~WINDOWS® environment is shown in FIG. 4. Thus, turning for a moment to FIG. 4, the computer can be reactivated by setting the address/object identifier of the interface to "1" for an active status (block 78). Since the reactivation request is preferably generated by the user, it is preferable that a message indicating that the insecure network is active is prompted or displayed on the computer (block 80). From this step, going back to FIG. 2, the process will be repeated from the step to wait for a predefined time (block 72). On the other hand, if there is no network reactivation requested (block 74), the process continues to the next step of determining whether the insecure network indicates an active status (block 82). In other words, the method 50 checks to determine whether the insecure network has already been deactivated. If not (block 82), the process will be repeated from the step of waiting for a predefined time (block 72).--

Please amend the paragraph beginning on page 10, line 19 as follows.

-- Turning now to FIG. 6, in order to determine whether the screen saver is active (Block 92), it must be first determined whether the current version of ~~Windows~~WINDOWS® is running on the computer 12 (block 94), which then separates into three different versions. If the version is not ~~Windows~~WINDOWS

NT®, the "FINDWINDOW" function is executed to find a "WINDOWS-SCREENSAVER" command (Block 96). If the "WINDOWS-SCREENSAVER" command is found (block 98), a determination of the screen saver being active is returned (block 100) back to the process shown in FIG. 2. Otherwise, a determination of the screen saver being not active is returned (block 102) to the process shown in FIG. 2.--

Please amend the paragraph beginning on page 11, line 4 as follows.

-- If it is determined that the current version of ~~Windows~~WINDOWS® is a NT version that is newer than 4.0 (block 94), a "SYSTEMPARAMETERSINFO" function is executed to find a "GETSCREENSAVERRUNNING" command (block 104). Similarly, if the "GETSCREENSAVER-RUNNING" command is found (block 106), a determination that the screen saver is active is returned (block 108) to the process in FIG. 2. Otherwise, a determination of the screen saver being not active is returned (block 110) to block 122 in FIG. 2.--

Please amend the paragraph beginning on page 11, line 12 as follows.

-- If it is determined that the current version of ~~Windows~~WINDOWS® is a NT version 4.0 or older (block 94), there is an attempt to open the desktop of the computer 12 where the screen saver is running on

(block 112). If the attempt to open the desktop is successful (block 114), a determination that the screen saver is active is returned (block 116) to block 122 in FIG. 2. Otherwise, it must be determined whether access has been denied by the program (block 118). If, in fact, access has been denied (block 118), a determination of the screen saver being active is returned (block 116). On the other hand, if access has not been denied (block 118), a determination of the screen saver being not active is then returned (block 120).--

Please amend the paragraph beginning on page 12, line 3 as follows.

-- Turning now to FIG. 7, to determine whether any active network process is currently running on the system in the ~~Windows~~WINDOWS® environment, the first step is to read an old number of received bytes and transmitted bytes (block 126), which is a number saved from the previous run through the process. If, however, this is the first time the process ~~is~~ has been run, the old number will be preferably zero. Next, the obtained address of the interface/object identifier (e.g., 1.3.6.1.2.1.2.7.?) must be changed to an address/object identifier (e.g., 1.3.6.1.2.1.2.10.?) (block 128) for obtaining or reading the number of bytes received during this process (block 130), which is then saved as a new number (block 132). Similarly, to obtain the number of bytes transmitted, the obtained address of the interface/object identifier (e.g., 1.3.6.1.2.1.2.10.?) is changed to an address/object identifier (e.g., 1.3.6.1.2.1.2.16.?) (block 134) for obtaining or reading the number of bytes

transmitted during this process (block 136). The obtained number of bytes transmitted is again saved as a new number (block 138). The old numbers of the received bytes and the transmitted bytes are then compared to the new numbers obtained (block 140). If the old numbers are equal to the new numbers (block 140), a determination that a network process is currently active and running is returned (block 142) to FIG. 2. If, however the old numbers do not equal the new numbers (block 140), a determination that a network process is currently active and running is returned (block 144) to FIG. 2.--